Delay Measurement Time Synchronization for Wireless Sensor Networks

Su Ping

IRB-TR-03-013

June, 2003

Intel Research Berkeley

Delay Measurement Time Synchronization For Wireless Sensor Networks

Su Ping
Intel Research Berkeley Lab

Abstract

A synchronized network time is essential for energy efficient scheduling, data fusion, localization and many other wireless sensor networks (WSN) applications. This paper studies the special issue of time synchronization in tiny sensor networking devices and presents a Delay Measurement Time Synchronization (DMTS) technique applicable for both single hop and multi-hop wireless sensor networks. DMTS is flexible and lightweight. For a single hop WSN of *n* nodes, it takes only one time broadcast to synchronize the network regardless the value of *n*. As aresult it adds minimum network traffic and is energy efficient, because radio communication is a significant source of energy-consumption in a WSN. For a multi-hop WSN of *n* nodes, DMTS requires *n* time message exchanges in total in order to synchronize the whole network.

DMTS is implemented in Berkeley motes within Tiny OS framework. It is a service available to TinyOS applications. Our test results show that DMTS achieves a time synchronization accuracy of 1 clock tick in single-hop WSNs. For a 2 hop WSN, the average time synchronization error is approximately 1.5 clock ticks.

DMTS scheme is currently used in several applications running on Berkeley motes to provide network timestamps and global scheduling.

1. Introduction

Wireless sensor network (WSN) is an emerging technology, which consists of very large number of tiny sensing devices, also called motes, distributed over physical space. Each device is capable of limited computing, radio communication and sensing. In the past couple of years, wireless sensor networks have found a wide range of applications such as environmental monitoring [4], robotic and object localization and tracking [5] [12].

Network time or time synchronization is essential for any communications networks, especially for wireless sensor networks. For example, data collected by sensors need global time stamps. When processing data collected from distributed motes, a global timestamp provides a foundation for merging individual sensor readings into a database. Synchronized network time is also essential for energy efficient scheduling and power management in wireless sensor networks. For example, in habitat monitoring applications, synchronized network time allows all the motes to shut down their radio and other peripherals, even the microprocessor to enter power saving mode simultaneously and later to return to its normal operation mode at a scheduled time simultaneously.

In this paper we present our study of network time issues in WSNs and existing time synchronization techniques. NTP is considered not suitable for wireless sensor networks [7]. While recognizing many advantages of Reference Broadcasting Synchronization (RBS), we notice the relative high network traffic overhead and consequently high energy consumption. Trading accuracy with energy efficiency, we propose an effective time synchronization approach named Delay Measurement Time Synchronization (DMTS) in this paper. As the name indicates, it is based the estimation of all delays involved in time synchronization message transfer path. It is

applicable for both single and multi-hop sensor networks. DMTS is flexible, lightweight and energy efficient. DMTS is implemented and tested in Berkeley motes [3] and integrated it into the Tiny OS framework [9] [3], an open source operating system for tiny wireless sensor network devices. The time synchronization accuracy achieved on Berkeley motes is up to hardware clock precision, which is 32 μs for single hop. For multi-hop DMTS, the worst-case is n times single hop time synchronization error. However, due to the cancellation of negative and positive time errors, the probability of worst-case scenario decreases while the number of hops increases. For a two hop WSN formed by Mica hardware, the average time synchronization error is 46 μs, or approximately 50% higher than that of single hop DMTS.

DMTS is currently used by several applications running on Berkeley motes for event time stamping and network event scheduling.

This paper is organized as following: Section 2 reviews existing time synchronization techniques. Special subsections are given to Network Time Protocol (NTP) and reference broadcasting synchronization (RBS) [7]. DMTS is described in section 3 followed by our test results in section 4. Section 5 discusses leader selection and global time scale, and section 6 service provision. Finally, we conclude in section 7.

2. Existing time synchronization techniques

2.1 NTP --- The Time Synchronization Scheme for Internet

Time synchronization is not a new research subject, though it is in wireless sensor networks. Extensive research has been conducted on how to transfer time or synchronize clocks that are distributed at different physical locations [10].

The time synchronization scheme widely adopted in traditional computer networks is the Network Time Protocol, NTP [13] [14]. A NTP client synchronizes its time with a NTP server using the server's time provided in a NTP packet and one half of the measured round trip time. In this way a NTP client can be synchronized within a few milliseconds error. We consider NTP is essentially a two-way time synchronization method, in which time is sent in both directions along the path. If the path is reciprocal or symmetric, one-way delay is estimated as one half of the round trip transit time. In wired computer network, there are too much stochastic factors in one-way time delay estimation, such as number of hops and traffic condition at each hop. Also IP network is not a broadcast media. These characteristics make one-way delay measurement and common-view time synch [10] unlikely candidates and two-way time synchronization as the choice of wired computer networks. It has proven itself over the years.

However, we argue that two-way time synchronization techniques, including NTP are not suitable in wireless sensor networks. The basic assumption of a two-way time synchronization method is that the time transfer path is reciprocal. Our study on path delays between a transmit node and a receiver node shows that this assumption is not true in wireless sensor networks. There are 5 delay factors in the signal transfer path between two nodes:

i ***Sender processing delay*** – This is the time elapsed from the moment a timestamp is taken to the point it is buffered in a mote's RF device.

ii ***Media Access delay*** – This is the duration for a timestamp message stays in

the radio device buffer. For TDMA system, this is the time spent waiting for a designated time slot. For CSMA system, this is the delay waiting for a clear channel to transmit.

iii ***Transmit time*** — this is the time for a radio device to transmit a time synchronization packet over a radio link. Since a packet have a fixed length and transmit speed for a given radio is a constant, this delay can be estimated. However, the estimation cannot be better than the sender-receiver synchronization error. On mica hardware platform [3], the maximum radio synchronization error is about 2 μs [1].

iv ***Radio propagation time*** – This is the time for a signal to propagate over the air to reach a receiver. Radio propagation speed is 300 meters per microsecond. Since the radio coverage of a wireless sensor network device is short and usually less than 100 meters. This error is negligible.

v ***Receiver processing time*** – Time consumed on receiver side to pass the received packet from RF device buffer to application module that is responsible for time synchronization.

Among these delays, transmit time and radio propagation time could be considered symmetric to the paths of different directions. The other three are not. Media access time is a key uncertainty. In addition to these, a time packet can be corrupted or lost along the path. If the MAC protocol retransmits the packet, the round trip time estimation error will increase significantly.

2.2 RBS

Elson proposed a Reference broadcasting synchronization technique [6][7][8] for wireless sensor networks. In RBS, a reference message is broadcasted. The receivers record their local time when receiving the reference broadcast, and then they exchange their recorded time. In this way, they have the knowledge of their time offset with each other. When each mote takes the average of its time offsets to all other nodes that have observed the same reference, a relative network time is achieved among all the receivers.

RBS is suitable for broadcast media as in a WSN and works best for pair-wise clock synchronization. The advantage of RBS is that all fluctuations in the transmitter side are cancelled out from time synchronization error. When the propagation time is ignored, the main error sources left are the processing time difference between receivers and the radio synchronization error between transmitter/receiver, which is typically a few microseconds [1]. As a result RBS enjoys high time synch accuracy as to 6 us for a single hop WSN using a system clock of 4MHz [7]. The disadvantage is that the number of message exchanges is high. For a single hop WSN of *n* nodes, it needs at least *n* messages to be exchanged between the nodes: One reference broadcast message from a reference sender and then one local time message from each receiver node. In the end the n-1 receivers are synchronized with each other, but not the reference sender. In a real wireless sensor network, the sender is likely a network node and therefore, it also need to be synchronized. To make this node equally synchronized with the rest of the nodes, another node needs to be the reference sender. Other n messages are broadcasted and a network time is achieved. This will result network traffic overhead and relative high energy consumption, which is a sensitive issue in a

WSN. A large number of message exchanges will also result in a longer convergent time, which measures how long it takes to synchronize a network.

3. DMTS

Having studied NTP and RBS, we felt the need to develop a more suitable time synchronization technique that avoids round trip time estimation, synchronizes sender and multiple receivers at the same time and require less number of message transfers than RBS. Other design considerations are scalability, energy consumption, computation cost and user application support.

A typical sensor mote has limited memory and limited processing power, therefore, low computation complexity and low memory usage is preferred.

One of the characteristics of sensor network is self-organization and dynamic behavior. The self-organization feature implies that the network topology may change from time to time. Therefore, we paid special attention to scalability and flexibility, which means being either adaptive or insensitive to changes in network topology.

Time synchronization creates additional network traffic. Bandwidth used by time synchronization traffic is a networking overhead. Heavy network traffic increases energy consumption by the RF device.

On top of these, we consider time synchronization as operating systems feature that should be integrated with the OS and provide an API to upper layers. We'll discuss each component of DMTS in the following subsections followed by test results.

3.1 Logical Clock

A wireless sensor device has typically two clocks: a system clock and a crystal oscillator external to the micro-control unit (MCU). For example, the popular Berkeley Mica mote has a 4 MHz system clock and an external watch crystal oscillator of 32,768 Hz [3]. The external oscillator can be configured at a scalable rate. Mica motes use a single channel, 916 MHz radio from RF Monolithics to provide a bi-directional communication path at 40 kbps. Media access protocol used is Carrier Sensing Multiple Access (CSMA) [12].

The system clock will stop when a device enters deep power saving mode and loses all of its time information. Therefore, it cannot be used for time keeping by itself. The external oscillator continues to operate when the MCU and other peripherals are powered off. It can be used to build a software logical clock for timekeeping and time synch.

A clock is measured by the following commonly used quality factors:

- ***Resolution*** --- The smallest possible increase of time a clock model allows. If a clock increases its value once per second, then its resolution is one second.
- ***Precision*** --- The random uncertainty of a measured value. Or the smallest possible increase of time that can be read.
- ***Accuracy*** --- It determines how close the clock is to a reference time.

Our logical time is represented by a 64 bits data structure. The unit of logical time is **$1/1024^2$ second**. Its resolution is limited by the frequency of the hardware crystal oscillator since it increments only when the associated hardware interrupt occurs. If the interrupt frequency is 1000 Hz, then the logical clock reading is updated every millisecond. This gives the logical clock a resolution of 1 ms. If there is no other measure or assistance in fine-grained clock reading, the precision of the logical clock will be the same as its resolution. Mica HW

clock interrupt interval may vary between 32 μs to 4 sec. When external hardware clock interrupt occurs, the logical time is incremented based on the current clock interrupt interval. For logical time read operation, the precision is one hardware clock tick, or 32 μs for best case in Mica motes.

A simple logical clock is a poor timekeeper. Its timing uncertainty is limited by the stability of the HW oscillator interrupt requests. Any change in the interrupt request rate causes the clock to gain or lose time. It is also possible for an ill-behaved software program to use the same hardware timer/counter for other purposes and change its interrupt rate. This could cause the clock to rapidly gain or lose time.

By reading hardware configuration at each interrupt time, our logical time is adaptive to hardware oscillator interrupt rate changes. This allows applications to modify clock rate at runtime without sacrificing the resolution and accuracy of logical clock. As a result, it also reduces the computation cost and power consumption of maintaining the logical clock. When a mote is in normal operation mode, we also use system clock to achieve higher precision time reading.

Logical time can be set to an external RTC time, in the unit of one decimal millisecond, using the following mapping equations:

$$t_l = 1000\, t/1024^2$$
$$t = 1024^2\, t_l/1000$$

3.2 DMTS

In DMTS a leader is selected as time master and broadcasts its time. All the receiver devices measure the time delay and set their time as received master time plus measured time transfer delay. As a result, all the devices that have received the time synchronization message can be synchronized with the leader. The time synchronization accuracy is bounded mainly on how well the delay measurements are along the path.

The 5 delays along a time transfer path are as presented in section 2.1. Now we discuss the means of measuring each of the delays and using Mica hardware platform as an example.

- Sender's processing time and MAC delay can be eliminated by taking a timestamp when a clear channel is detected.
- Transmit time can be divided into two parts: time to transmit preamble and start symbols and time to transmit data. Preamble and start symbols maybe transmitted at the different speed. If we know the transmit speed and number of bits transmitted, we can estimate this delay to an accuracy approximately equivalent to one half of the time spent in transmit a single bit.
- Receiver processing time can be measured using receiver's local timestamps. If we timestamp a time packet at it arrival time, and give it another timestamp before adjusting the receiver's logical clock. Receiver processing delay is the difference between the two timestamps. The measurement accuracy is limited by the resolution of the local clock used for the timestamps.

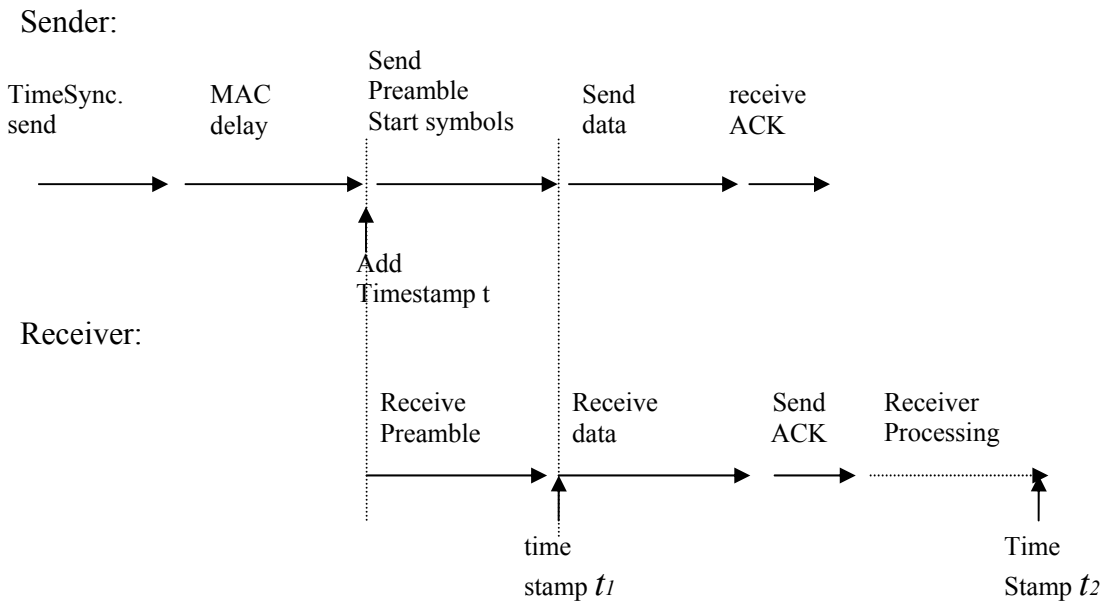


Figure. 1. Time transfer path in a Mica mote.

In Mica platform, the sender's and receiver's radio device will synchronize with each other in the end of the start symbol. If a receiver take a local timestamp when its radio synchronized with the sender's radio, and take another local timestamp when processing the time message, it then has measured both the data transfer time (excluding preamble and start symbols) and receiver side processing delay.

Figure 1 shows the time line of transfer a time message from one node to another in mica hardware platform. When the radio propagation delay is neglected, the total delay t_d can be measured as:

$$t_d = t_e + (t_2 - t_1)$$

Where t_e is the estimated time to transmit the preamble and start symbols, t_2 and t_1 are receiver timestamps. Since a radio device has a fixed transmit rate, for example, Mica radio transmit preamble and start symbols at the speed of 20 kbps, t_e is a fixed delay and can be calculated as

$$t_e = n\tau$$

where n is the number of bits to transmit and τ is the time to transmit one bit over radio.

In DMTS, a time synchronization leader sends a time synchronization message with its timestamp t, which is added after MAC delay and a clear channel is detected. The receiver measures the path delay and set its local clock to t_r.

$$t_r = t + n\tau + (t_2 - t_1)$$

The receiver is then synchronized with the leader. The lower bound of DMTS is the radio device synchronization accuracy, and the upper bound is the precision of local clock. In Mica platform, a receiver's radio device can synchronize with a sender with an accuracy of 2 μs error. The precision of the logical clock is 32 μs. As a result, time synchronization errors in a single hop WSN formed by Mica motes should be between 2 μs and 32 μs.

Since only one time signal transfer is required in DMTS to synchronize all nodes within a single hop. This method is energy efficient. It is also lightweight because there are no complex operations involved. From the receivers' points of view, the sender's time signal is a common-view timestamp, hence all the receivers synchronize with each other better than with the sender. This is mainly because the transmit time estimation error is cancelled out.

3.3. Multi-hop DMTS Algorithm

DMTS can be extended to multi-hop sensor network. If a node knows that it has child/children, it broadcasts a time signal after it adjusts its own time. This situation becomes a single hop time synchronization problem with a second level leader. However, in some WSNs, a node has no knowledge of its child/children. To tackle this issue, we propose the following multi-hop time synchronization algorithm.

A leader selection algorithm is used to select a time master. The concept of time source level is used to identify the distance from the master to another node. A time master is of time source level 0. A node that has synchronized with the master is level 1 time source. A node that synchronized with a level *n* node will have a time source level *n*+1. The root node will periodically broadcast its time. So are the synchronized nodes. Each node that has been synchronized directly or indirectly with the master will broadcast its time once and only once for a given time sync period. When receiving a time signal, a node will check the level of the time source. If it is from a source of lower level than itself, it accepts the time. Otherwise, it silently discards the signal. This algorithm guarantees that master time will be propagated to all network nodes with limited number of broadcast, which is equal to the number of nodes. It also warrants the shortest path to the time master, or the least number of hops, because a node always selects the node that is closest to the time leader as its parent.

In theory, the maximum error in such a time synched *N* hop WSN is *N* times simple hop time synchronization error as described in section 4.2, where *N* is the number of hops in a time synchronization network. In reality, this worst case rarely happens because the error over multiple hops would cancel each other.

4. DMTS Test Results

DMTS is implemented under TinyOS framework on Mica hardware platform and tests are conducted for both single hop and 2 hop networks. In all the tests, the node of smallest ID is selected as leader.

1078 initial time synchronization tests using DMTS were conducted. Initial time synchronization error is measured by reading logical time from the time leader and the receivers after time synchronization message is sent/received and calculating the absolute time difference between the leader and a receiver. The error distribution is show in figure 2 in the format of a histogram over 300 bins. The X-axis represents logical time difference between the leader and a receiver in the unit of binary microsecond.

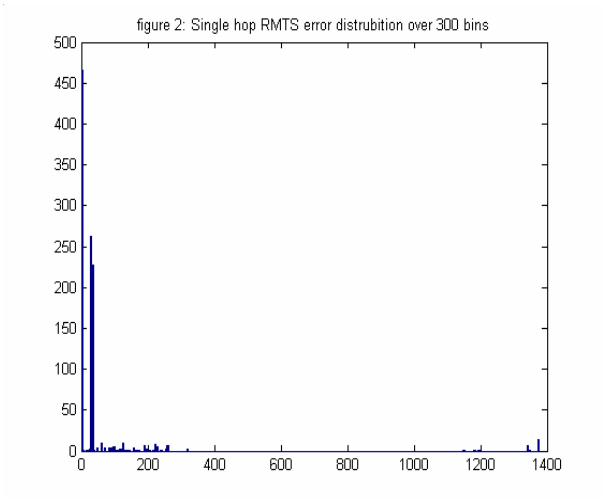


Figure 2: Single Hop DMTS error distribution.

There are two peaks in the distribution, one in the bin with an error smaller than 5, and the other is around 32 microseconds. This is

because our hardware oscillator runs at 32,768 Hz, which makes 32 binary microseconds the smallest unit to increase our logical time. As a result, reading logical time at different phase of a clock cycle result in 32 microseconds error. There are a small number of cases that DMTS initial synchronization error is as high as 1400 binary microseconds. We think that in these cases context switch happened due to HW signal or interrupt. There is a small window between a receiver taking timestamp t_2 and setting its local time to the estimated time t_r. This includes the time spent in calculating t_r and time for setting the receiver's logical time. This window adds extra error. If these processes are executed within a critical region and with interrupt disabled, then the extra delay can be estimated based on the CPU speed and the number of instructions involved. If this is not true, as in our test code, then the extra delay is not a constant since it maybe interrupted and the added delay depends on the length of the interrupt service routine.

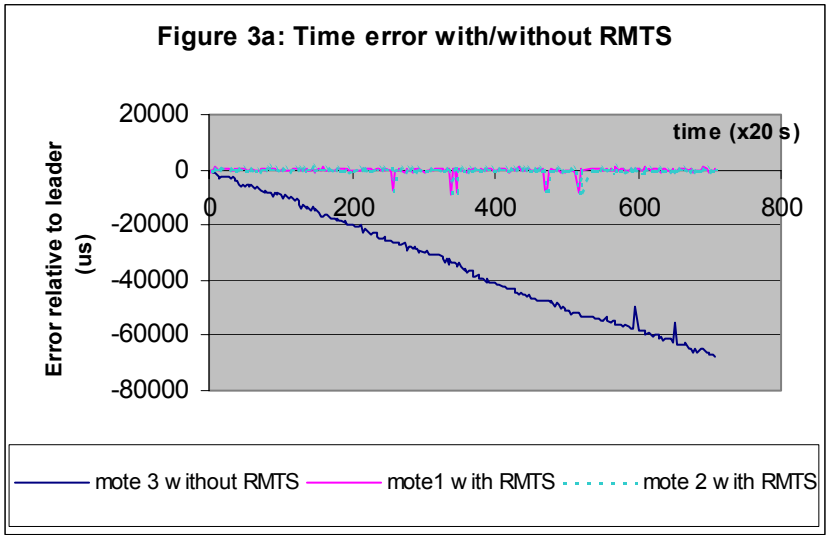
A smaller number of 2 hop initial time synch experiments were conducted. The majority of them have an error within 32-64 us. The average error is 46 us when the suspected cases of missing a interrupt are excluded.

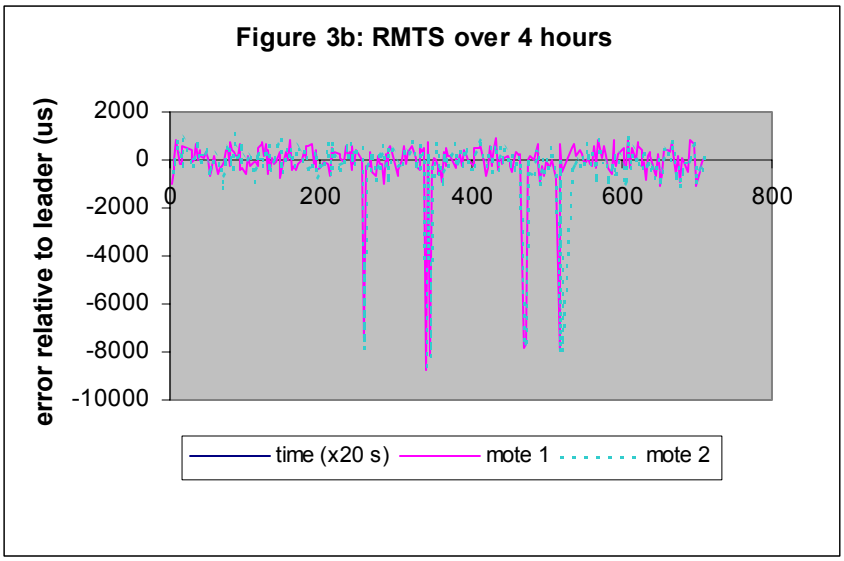
We also tested DMTS in a 2 hop real WSN for over 4 hours. In this case, node 0 is selected as time master. It sends a time synchronization message every 3 minutes. Another node, as an event trigger, broadcasts a time request every 20 seconds.

When receiving a time request, a node, regardless it is a time master or slaves, reads its time and sends the time reading to the node's UART, which is connected to a local network. Node 3 does not have DMTS running and is not synchronized with the rest of network. It shows a drift of -5 PPM relative to node 0. Node 1 and 2 are synched by multi-hop DMTS. Node 1 is one hop away from the time master and node 2 is 2 hops away. The test results are shown in figure 3a and 3b.

The majority of the time synch error is within 2 ms. We expect the time error to be within 4 ms since the frequency variance of the HW clock is ±20 PPM. However, at certain point, the error is around 8 ms. If we change the hardware clock interrupt interval, for example to 16 ms, then the error value changes from 8 ms to 16 ms. We believe that this is caused by a miss in clock interrupt in Mica hardware.

Figure 3a: Time error with/without RMTS





5. Leader selection and global time scale

A global time scale for a DMTS synchronized WSN can be achieved by connecting a global time source to the selected time synchronization leader.

Any leader selection algorithm may apply in theory. However, we consider simplicity as a key requirement. It is practical to have a designated leader and only run the leader selection algorithm when it is not present for a given period of time.

A leader selection algorithm can be very simple, such as select the node of smallest ID. In practice it works best to have base station node as the default time root or master. The reasons behind are (a) base station is well attended relative to other nodes; (b) it is more likely to have a constant source of power supply other than small capacity batteries; (c) since base station is normally connected to a larger networked computing device, it is more likely for it to access a RTC time or other form of global time.

6. Service provision

Time synchronization as an integrated kernel component must provide services to application and higher-level kernel modules. Otherwise, it is meaningless. Out time synchronization module provide mainly two significant services, (a) Time reading or stamping and (b) Absolute timers for network event scheduling. The services are provided in the format of user application interfaces. The rational behind providing absolute timer services is that one cannot have a synchronized event using other OS service such as a typical timer. Times provided by a typical embedded kernel are relative ones. It is relative to a certain point at time domain. When a user application start a relative timer, what it asks from the kernel is: signal me n seconds from now. Without an absolute time, user applications can not specify now, therefore, it is impossible to schedule a synched event over a network.

7. Conclusions

DMTS is energy efficient time synch approach for wireless sensor networks, because only one time signal transfer is required to synchronize all nodes in a single hop. It is also lightweight since there is no complex operation involved. From receivers' points of view, the leader's time signal is a common-view timestamp, hence

all the receivers synchronize with each other better than with the sender. DMTS can synchronize a single hop WSN with 1 clock tick error, which is 30 μs in our test hardware, and a 2 hop WSN of 1.5 clock ticks in average. Our run-time synchronization test over 4 hours in a 2 hop network shows that it can be used to overcome drifts and keep network time synchronized within a couple milliseconds.

Although DMTS is implemented and tested on Berkeley's Mica hardware platform, the technique is general and can be implemented in other wireless sensor network devices, which may have a difference real-time embedded operating system.

In comparing with RBS, DMTS is lighter in computational cost, especially when a global time scale is required. It is also more energy efficient due to the smaller number of message broadcasting required. However, its accuracy is lower than RBS. We traded off complexity and energy efficiency to accuracy.

On the application front, one significant difference between our work and other time synchronization research is that we apply our technique to a relatively low resolution, low frequency external clock. We also emphasize service provision to upper layer modules in a system.

Reference:

[1] J. Hill and D. Culler. Wireless embedded sensor architecture For system-level optimization. Technical report, U.C. Berkeley, 2001.

[2] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler and K. Pister. System architecture directions for networked sensors. *Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX)*, pages 93–104, Cambridge, MA, USA, November 2000. ACM.

[3] http://webs.cs.berkeley.edu/tos/

[4] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, Wireless Senor Networks for Habitat Monitoring, *2002 ACM International Workshop on Wireless Sensor Networks and Applications* September 28, 2002, .Atlanta, GA.

[5] L. Girod and D. Estrin. Robust range estimation using acoustic and multimodal sensing. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2001)*, March 2001.

[6] J. Elson and D. Estrin. Time synchronization for wireless sensor networks. In *Proceedings of the 15th International Parallel and Distributed Processing Symposium (IPDPS-01)*. IEEE Computer Society, April 23–27 2001.

[7] J. Elson, L. Girod and D. Estrin , Fine-Grained Network Time Synchronization using Reference Broadcasts. *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002), Boston, MA. December 2002.*

[8] J. Elson and K. Römer, Wireless Sensor Networks: A New Regime for Time Synchronization. *Proceedings of the First Workshop on Hot Topics In Networks (HotNets-I), Princeton, New Jersey. October 2002.*

[9] J. Hill, A Software Architecture Supporting Networked Sensors, Masters thesis, December 2000

[10] J. Levine, Introduction to time and frequency metrology, *Review of Scientific Instruments, June 1999, pp. 2567—2596*

[11] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–65, 1978.

[12] A. Woo, D. Culler, A Transmission Control Scheme for Media Access in Sensor Networks, *Mobicom 2001, July 2001, Rome.*

[13] D. L. Mills. Internet Time Synchronization: The Network Time Protocol. In Zhonghua Yang and T. Anthony Marsland, editors, *Global States and Time in Distributed Systems*. IEEE Computer Society Press, 1994.

[14] D. L. Mills. Precision synchronization of cmputer network clocks. *ACM Computer Comm. Review*, 24(2):28–43, April 1994.

[15] D. L. Mills. Adaptive hybrid clock discipline algorithm for the network time protocol. *IEEE/ACM Transactions on Networking*, 6(5):505–514, October 1998